TAB The Electronics Authority

DISK Included

PROGRAMMING AND CUSTOMIZING THE PIC MICROCONTROLLER

MYKE PREDKO

Contents

Acknowledgments *ix*

Introduction *xi*

Microcontroller applications *xi*
PIC resources *xiii*
Conventions used in this book *xiv*

1 Microcontrollers *1*

Different types of microcontrollers *1*
The PIC microcontrollers *2*
The Parallax Basic Stamp *3*

2 PIC hardware features *5*

Different types of PICs *5*
Device packaging *6*
Control store types *9*
Reset *10*
System clock/oscillators *11*
Hardware and file registers *13*
Interrupts *15*
Timers *15*
Digital I/O *16*
Serial digital I/O *17*
Analog I/O *20*
Slaved devices *21*
In-System Programming *22*

3 The PIC processor architecture *25*

The CPU *25*
The STATUS register *28*
Register addressing *31*
The Program Counter *34*
Peripherals *39*
Interrupts *39*
Parallel input/output *41*
Timers *43*
TMR1, TMR2, and the Capture/Compare/PWM (CCP) modules *47*
PIC input/output control registers *49*
CPU Program Counter stack *58*
Sleep *60*
Configuration fuses *61*
PIC 17C4*x* *architecture* *62*

4 The PIC instruction set *67*

Register arithmetic instructions *68*
Immediate arithmetic operators *80*
Execution change operators *84*
Microcontroller control operators *86*
Bit operators *91*
High-end (17C4*x*) PIC instructions *93*

5 Application design *97*

Power input *97*
System oscillators and clocks *98*
Reset *101*
Interfacing to external devices *104*
Interrupts *111*

6 PIC application software development tools and techniques *117*

Software development tools *117*
Creating and supporting your code *118*
Languages *119*
Version-support tools *124*
Simulators *125*
MPLAB *125*
Going through the files *126*
Programming styles *138*
Debugging your program *148*
In closing *150*

7 Programming PICs *151*

Programming *151*
Available programmers *153*
PROG39: The YAP *156*

8 Experiments *165*

Tools *165*
Experimenting with the PIC *169*
PROG1: Register addressing *169*
PROG2: Register bank addressing *172*
PROG9: The STATUS register *174*
PROG43: Jumping around *178*
Data *178*
PROG4: Variable manipulation *179*
PROG5: Simulating a variable array *182*
PROG6: Simulating a stack for arithmetic operations *184*
Subroutines *187*
PROG7: Passing data through registers *187*
PROG8: Passing parameters on a stack *189*
PROG10: Blowing up the Program Counter stack *190*
Table data *192*
PROG14: Table calling and placement *192*
PROG44: Tables longer than 256 entries *194*
PROG12: Execution control state machines *196*
Playing with the hardware *199*
PROG2: Programming the PIC/turning on a light-emitting diode (LED) *200*
PROG16: Polling an input bit *203*
PROG13: Changing an output bit value inadvertently *204*
PROG17: Power/decoupling problems *207*
PROG17: Different oscillators *209*
PROG17: The watchdog timer *210*
PROG21: Reset *211*
PROG18: Register power-up values/button press with debounce *213*
PROG47: Sleep *218*
I/O with interrupts *220*
PROG45: Timer/interrupt handler with context saving *221*
PROG46: Debouncing inputs *223*
Analog input/output *227*
PROG20: Measuring resistance values *227*
PROG31: Resistor ladder output *230*
Closing out the experiments *236*

9 Projects 237

Blinking lights for Christmas decorations *238*
PROG35: Serial-LCD interface *241*
PROG25: I/R tank *247*
PROG36: Addendum to the I/R tank *258*
PROG32: Electronic thermometer with 7-segment LED displays *263*
PROG33: Marya's Toy—addendum to the LED thermometer *267*
PROG38: The MIC *269*
PROG40: Keyboard interface *274*
PROG41: Servo controller *278*
PROG41: Addendum—SimmStick servo controller *282*

10 Emulators *285*
The PICMaster *285*
PROG42: The Emu *286*

APPENDICES

A Hints *293*
Software hints *293*
PIC execution hints *294*
Interfacing hints *295*

B 16-bit operations *297*
Defining 16-bit numbers *297*
Increments and decrements *298*
Addition/subtraction of constants *298*
Addition/subtraction of other variables *299*
Other operations on constants and variables *300*
Comparisons with 16-bit variables *300*
Multiplication *302*
Division *303*

C Reuse, return, recycle *307*
Useful routines *307*
Include files *314*

D PIC resources *327*
Microchip *327*
Contacting me *327*
The PICLIST *328*
My favorite PIC Web sites *328*
Periodicals *329*
Web sites of interest *330*
Suppliers *331*
PIC-specific companies *333*

Epilogue *347*

Index *348*

About the Author *353*

Software and License Information *354*

Introduction

As you grow older, most things that you remember will become cloudier and less clear ("I dated who?"), while others will stay as vividly in your memory as though they just happened. These snippets of what happens around us are vitally important in shaping us as human beings. They are known as "Significant Emotional Events" and, as a psychology professor I once had described, they are largely responsible for defining who we are.

I always felt that I received a SEE (as a Significant Emotional Event is known to psych profs) when I first opened up a copy of the IBM Personal Computer Technical Reference and saw the schematic for the keyboard section of the PC. The keyboard was simply a switch matrix wired to a single box, labeled "18048," which was in turn wired to the PC itself. Finding a copy of Intel's "Microcontroller Data Book," I discovered the 8048 was a chip with a microprocessor, memory, interrupt control, and input/output—an analogous computer system that was communicating to the main part of the PC, which took up 20 or so pages of schematics!

Reading through the databook, I marveled at how much the little creatures could do. My mind was abuzz with potential applications, and I started listing them out on paper.

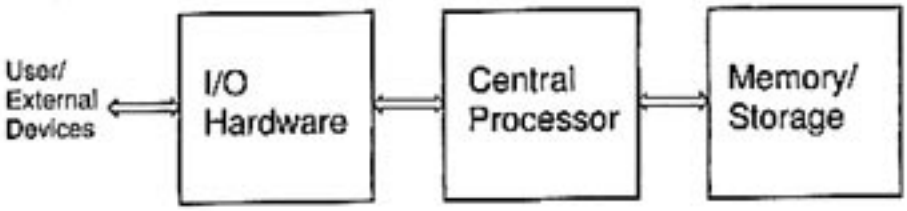
That was a long time ago, and still my philosophical wonderment for microcontrollers hasn't died down. In the years since first looking at that schematic, I've had an opportunity to design a few applications using microcontrollers and learn what their capabilities are.

So what is a microcontroller? If you remember your first instruction in computer science, you'll remember the diagram shown in Fig. I-1.

A microcontroller contains these three basic functions (along with a few extra) all contained on one chip.

Microcontroller applications

Where can you find a microcontroller? Well, just about anywhere you look. If you could pick up your car and shake it, I'm sure a number would fall out. In the car, they



I-1 Basic computer block diagram.

are used for controlling the engine, the brakes, the radio, and even the seats. Your kitchen appliances have so many that the joke of the talking toaster isn't all that far-fetched. If you remember back about 15 years to the Air Canada Boeing 767 that glided down to a drag racing strip after running out of fuel, the aircraft had 69 microcontrollers that all stopped working when the engine-driven generators stopped.

After this list, you're probably thinking that something used in a toaster would be radically different from what is used in an airliner. Actually, they aren't.

While there is a myriad of different devices out there, they can all be modelled using the basic diagram in Fig. I-1. Learning about one chip, will give you understanding as to how all microcontrollers are designed into applications.

The decision to determine which microcontroller to use is answered by making a list of requirements that you have for the application. Before choosing which type of microcontroller to use, a list of requirements should be made. This list would then be matched to various microcontroller descriptions, and the one that was best suited could be identified easily. However, before you are able to do this, you should really have an idea of what microcontrollers can do.

This turns into a chicken and egg question when you ask which microcontroller should you start with? If you go out and search manufacturer's Data Books, you will be bombarded with different types of devices with a myriad of different features. When you compare what each manufacturer offers, you'll find that, when the total range of the devices is examined, you can pretty much find similar features across the product lines. The decision on what device to use will really boil down to such points as cost, availability, support, and what you are familiar with.

I've picked a family of devices that contains features that are fairly common to devices in general and has some advantages to allow you to experiment with it easily and without breaking your budget. The tools necessary for doing all the applications in this book should be sitting on your workbench right now.

The microcontroller family discussed in this book is the "PIC" Series by Microchip Inc. (Microchip's address can be found in appendix D.) The devices contain a RISC (Reduced Instruction Set Computer) Processor, anywhere from 256 bytes to 16K of Instructions of Program Space, tens to hundreds of bytes of Variable RAM, and some number of I/O bits (which can have additional advanced features). Like most microcontrollers, the features that make this device so attractive for use in a very wide range of applications include it's robust design with regards to Power, Clocking, and Reset.

For this book, I've primarily focused on the PIC16F84, which has Electrically Erasable PROgrammable Memory ("EEPROM") for program storage. This allows simple changes to the contents of memory without having to invest in ultra-violet erasers. The software development tools used in this book are available free of charge from Microchip.

Actually programming the devices can be accomplished either by various manufacturers' programmers, or by using the Programmer circuit that is included in this book. The programmer in this book has many of the features of other low-cost programmers that are available, but with the important difference that it is capable of supporting a wide variety of different hosts (and not just the IBM PC). I have included a number of sources for programmers in appendix D.

Also included in this book is the design for a simple PIC emulator. As I have designed the emulator, I can really see how useful it can be for faster development and debugging of applications. Like the programmer, the emulator has a simple RS-232 Interface that allows its use with a wide variety of hosts. While it is very low-cost, there are two major limitations on its usefulness in helping to debug applications: The emulator is relatively slow and is unable to use symbolic information from the source code. However, even with these limitations, it is an excellent tool to help understand how the PIC works.

As you go through this book, I've tried to build knowledge in a "bottoms up" format, in the following order:

- The features of the microcontroller
- Application design notes and rules
- Investigating the microcontroller
- Sample applications and projects
- Turning the reader loose

Please don't think that I've written a book on just the PIC microcontrollers (and the 16F84 specifically). I've tried hard to provide a framework that anybody can use to investigate a microcontroller or choose the appropriate one for a given application. I hope you don't feel I've given your favorite microcontroller the short shrift, because that's not my intention at all with this book. What I've tried to do is pass along some of the knowledge I've picked up over the years along with a few of the fun things I've created. I've tried to keep this book (well) below the level of a graduate engineer.

As this book went to press, the original EEPROM based PIC (the PIC16C84) became "obsolete". This was a mainstay device for learning about the PIC and developing first applications on. The 16C84 has been replaced by the 16F84, which has more file registers and some differences in the configuration fuses, by Microchip and will be the primary device used in this book.

This book is not meant to explain electrical engineering and computer science basics. Concepts and ideas will probably be touched upon in this book that aren't fully explained or obvious. If you do discover something that you are not sure of, you can find full explanations in your local library.

I hope you find this book useful, both in terms of learning about the PIC, as well as giving yourself an idea of what microcontrollers can do in general.

PIC resources

In this book, you will see numerous references regarding where to get software and information. The latest revisions of this data can be found on the Internet. I have tried to ensure that only resources that are on the 'Net are used in this book because of the Internet's common and consistent manner of accessing data.

All of the tools you will find listed in the appendices are for PC-DOS machines. Microcontrollers generally do not require a powerful development system, and the PC is well documented and easy to use. There are Internet sources with source code for the various tools that you might be interested in porting to other platforms.

While there are numerous sources for assemblers, compilers, simulators, and programmers, I have selected and included Microchip's PC-DOS command-line development tools for this book ("MPASM" and "MPSIM") along with the programmer and emulator. This means that the book contains all of the tools required to develop (18-pin) PIC applications.

At the end of this book, I have listed a number of sources for information and products. I consider the "list groups" to be the most important resource you can have. "List groups" are informal programs that distribute Internet e-mail to a list of people. This e-mail usually consists of questions and answers, information, and product updates. I have found this to be truly the best single resource that you can have for PIC information.

Conventions used in this book

KΩ—1000 ohms

µF—Microfarads

ms—Milliseconds

µs—Microseconds

0x0*nnn*—Hex number

0b0*nnn*—Binary number

nnn—Decimal number

AND, OR, XOR—Bitwise logical operations

_Label—Negative active signal/bit

Register.Bit—Single bit in a register

`Monospace font`—Code examples

Electronics

$34.95 U.S.A.

FOR COMPLETE, PRACTICAL INSIGHT INTO THE CHIP THAT IS QUICKLY BECOMING THE MICROCONTROLLER OF CHOICE THROUGHOUT THE WORLD

The use of Microchip's PIC family of microcontrollers is growing rapidly worldwide. This tutorial/disk package is unique in providing you with a complete understanding of the PIC chip *and* all the information needed to design and debug tailor-made applications using the PIC family of devices.

Programming and Customizing the PIC Microcontroller details the features of the PIC microcontroller and how different devices are accessed and controlled. It shows what happens within the PIC when an instruction is executed and how PICs can be interfaced with external devices. The heart of the book includes:

- 12 complete projects which illustrate the PIC's capabilities
- Over 30 experiments to reinforce your grasp of how the PIC works
- A discussion of how the devices are electronically programmed
- An explanation of the tools used for developing code for PIC applications
- Suggestions for debugging and easier code development.

A diskette augments the book by offering all the Microchip software tools needed to program the PIC, sample application programs, and reusable code. Extensive appendixes include additional resources that can help you in developing and debugging PIC applications as well as tips on how to avoid typical application problems.

Visit us on the World Wide Web at
www.tabelectronics.com

Cover: Square One Design
Cover Illustration: Brad Goodell

McGraw-Hill
A Division of The McGraw-Hill Companies

P/N 052583-8
PART OF
ISBN 0-07-913646-X